IBIS Interconnect SPICE Subcircuits Specification
(IBIS-ISS)

Draft 0.4~~3~~
June 14, 2010

Formatted: Heading 1

1. Statement of Changes

- Eliminated or reduced use of terms: keyword, netlist, command
- Defined or refined existing definitions for key concepts: statement, token, delimiter
- Added table of contents and imposed formatting (as defined under "Conventions") to clarify relationships and hierarchies
- Added skeleton text for .subcircuit and .model definitions
- Standardized (for the most part) fonts, typefaces and other visual aspects of the document per "Conventions" section
- Removed portions of the document conventions that were unused
- Reordered some minor portions of the document to better accommodate the flow (more like a narrative)
- Clarified existing examples, particularly for comments and expressions
- Added parameter description text to support strings
- Clarified line continuation text

Formatted: Font: Arial, 12 pt

Formatted: Font: Arial, 12 pt

Formatted: Font: Arial, 12 pt

Formatted: Font: Arial, 12 pt

Formatted: Font: Arial, 12 pt

Formatted: Font: Arial

Contents

Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font

Formatted: Default Paragraph Font

Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font

Formatted: Numbered + Level: 1 + Numbering Style: 1, 2, 3, … + Start at: 1 + Alignment: Left + Aligned at: 0.25" + Indent at: 0.5"

1. Overview

The IBIS Open Forum, in order to enable easier data exchange between users of signal/power integrity simulation and physical layout/routing software tools, is issuing a generic netlist format, to be called “IBIS Interconnect SPICE Subcircuits” (IBIS-ISS).

This format is similar in structure and major functions to the SPICE (Simulation Program with Integrated Circuit Emphasis) nodal syntax developed at the University of California at Berkeley and since implemented in various forms by individual software tool vendors. IBIS-ISS is the first industry-wide attempt to standardize SPICE subcircuit representation.

This version of IBIS-ISS is based on a subset of HSPICE ®, used with permission from Synopsys, Inc. HSPICE is a registered trademark of Synopsys, Inc.

2. Goals and Scope

The syntax of IBIS-ISS is intended to:

- describe interconnect structures (such as PCB traces, connectors, cables, etc.) electrically, for analysis in a signal integrity and/or power integrity context
- describe the arrangement or topology of interconnect structures, as they relate to each other and to active devices in a system

To these ends, IBIS-ISS includes support for:

- elementary circuit elements (resistors, capacitors, inductors)
- transmission line elements (lossless and lossy)
- frequency-domain network parameters (e.g., S-parameters)
- parameter/variable passing to elements and subcircuits
- dependent sources
- string-based node naming
- user-defined comments
- abstraction through modular, user-defined subcircuit definitions

IBIS-ISS does NOT include or cover:

- descriptions of complete netlists intended for input “as-is” to simulation tools
- model formats or “process cards” for active devices (e.g., diodes, transistors)
- independent sources
- controls or options for any simulation engine (e.g., precision, algorithm selection)
- simulation or analysis types (e.g., DC, transient)
- sweep or run control (e.g., Monte Carlo)
- geometrical descriptions for field solvers
- support for other kinds of data extraction/export (e.g., S-parameter generation)
- measurement, printing or probing
- encryption support

3. Best Practices

Scaling of interconnect subcircuits may give different results between different simulators and should be avoided.

Global parameters may give different results between different simulators and should be avoided.

Exponent range shall be limited to between e-60 and e+60.

For maximum compatibility, IBIS-ISS does not support the “X” (Meg) scale factor..

A name field shall begin with [a-z] or [A-Z]. The remaining characters of a name field shall be limited to

- [a-z], [A-Z], [0-9], ~!@#%&_<>?[]|:;

Parameter names shall begin with [a-z] or [A-Z], and the remaining characters shall be limited to

- [a-z] or [A-Z], [0-9], ! # $ % [ ] _

While a Parameter may be defined in more than one .param statement within a subckt, this practice is best avoided.

Node names shall either be all numeric [0-9], or be a Name Field.

4. Conventions

The following typographical conventions are used in IBIS-ISS. Note that these may be combined (e.g., Courier font in bold type).

Convention	Description
`Courier`	Indicates statement syntax.
Italic	Indicates a user-defined value, where a specific text string will replace the italics shown in an actual IBIS-ISS file (e.g., `R`*`xxxx`* is a generic representation of a resistor element name, such as `Rname`).
Bold	Indicates verbatim text in syntax and examples
`[]`	Denotes optional tokens
...	Indicates that tokens of the same type may be added as appropriate to the element structure: pin1 pin2 ... pin
|	Indicates a choice among defined alternatives, such as low | medium | high
+	Indicates a continuation of a statement across input lines. Note that continuation may only be used between tokens and shall not split any single token across lines.

5. Input Structure and Data Entry

This section describes the input file and structures for representing input data.

1. Input File Guidelines

An input file consists of a collection of statements describing a portion of a complete circuit. This input file is intended for inclusion in a larger netlist or description of a complete circuit, to be used by a simulation tool.

An input filename may be up to 1024 characters long. The input file shall be in ASCII format (insert IEEE or ANSI definition here). The input file shall not be in a binary, packed or compressed format.

2. Statements and Tokens

A statement in IBIS-ISS is a text string consisting of tokens and delimiters. An IBIS-ISS file may contain multiple statements (the number of statements is not limited by the IBIS-ISS definition, but may be limited by the computer architecture and/ or operating system used to process the file).

Any individual input statement may be up to 1024 characters long.

Statements in an input file may appear in any order.

Any valid string of characters between two token delimiters is a token.

For the purpose of this specification, statements are grouped into the following types:

- Element instances
- Parameter definitions
- File includes
- Subcircuit definitions
- Model definitions
- Comments
- Subcircuit ending statements

Subcircuit ending statements, subcircuit definitions, model definitions, parameter definitions and file includes all begin with the dot (.) character.

The specific syntax of the above statement types are described in the sections below.

- IBIS-ISS ignores differences between upper and lower case in input statements, except in quoted filenames.

- To continue a statement across multiple lines, the plus (+) sign shall be used as the first non-numeric, non-blank character of each continued line. The + sign shall be used only between tokens and token delimiters and never to split tokens.

- Tokens with extended length (such as paths and expressions) may span multiple lines using a single whitespace character followed by a backslash (\ ) or a double backslash (\\ ) without leading whitespace at the end of the line containing the token to be continued on the following line. Note that quoted strings may only be continued using the double backslash (\\ ) sequence.

- ~~Parameters are used in two contexts.~~
  - ~~Parameters in parameter definition statements are strings, defining names to be used as variables which are assigned specific values by the statement. These values may be numeric, strings defining an expression or equation, or strings matching parameters defined elsewhere.~~
  - ~~Parameters may also appear in element instances, model definitions and subcircuit definitions. These parameters may be user-defined or may use names pre-defined by the syntax of the element.~~

- ~~Parameter names must begin with an alphabetic character, but thereafter can contain numbers and some special characters.~~

~~Curly braces ({ }), are interpreted as square brackets ([ ]).~~

- ~~Names are input tokens. Token delimiters must precede and follow names.~~
- ~~Names can be up to 1024 characters long and are not case-sensitive.~~
- ~~Do not use any of the time keywords as a parameter name or node name in your netlist.~~
- The following symbols are reserved operators:

  ( ) = " ‘

These symbols shall not be used as part of any parameter or node name.

Formatted: Bulleted + Level: 1 + Aligned at: 0.25" + Indent at: 0.5"

Formatted: Font: Helvetica

Formatted: Font: (Default) Helvetica

Formatted: Font: (Default) Helvetica

Formatted: Font: Helvetica

Formatted: Font: Helvetica

Formatted: Font: (Default) Helvetica

Formatted: Font: (Default) Helvetica

Formatted: Indent: Left: -0.25", Bulleted + Level: 1 + Aligned at: 0.25" + Indent at: 0.5"

Formatted: Indent: Left: 0.54", Hanging: 0.29"

Formatted: Indent: Left: -0.25", Hanging: 0.25"

Formatted: BulletPrev, Indent: Left: -0.25", Hanging: 0.25", Bulleted + Level: 1 + Aligned at: 0.69" + Indent at: 0.69"

Formatted: Font: (Default) Helvetica

Formatted: Font: (Default) Helvetica

Formatted: Font: (Default) Helvetica

Formatted: Font: (Default) Helvetica

Formatted: Indent: Left: 0.25", Hanging: 0.29"

Formatted: Font: (Default) Helvetica

Formatted: Indent: Left: 1"

Formatted: Indent: Left: 0.5"

3. Special Characters

The following table lists the special characters that may be used as part of node names, element parameter names, and element instance names. For detailed discussion, see the appropriate sections in this chapter.

Note:

To avoid unexpected results or error messages, do not use the following mathematical characters in a parameter name in IBIS-ISS: * - + ^ and /.

Table 4 *IBIS-ISS / Netlist Special Characters*

Special Character “Legal anywhere”=first character or any position in name “Included only”=any position except first character		Node Name	Instance Name (cannot be the first character; element key letter only)	Parameter Name (cannot be the first character, element key letter only)	Delimiters
~	tilde	Legal anywhere	Included only	Included only	n/a
!	exclamation point	Legal anywhere	Included only	Included only	n/a
@	at sign	Legal anywhere	included only	Included only	n/a
#	pound sign	Legal anywhere	Included only	Included only	n/a
$	dollar sign	Included only (avoid if after a number in node name)	Included only	Included only	In-line comment character
%	percent	Legal anywhere	Included only	included only,	n/a
^	caret	Legal anywhere	Included only	included only (avoid usage),	“To the power of”, i.e., 2^5, two raised to the fifth power
&	ampersand	Legal anywhere	Included only	Included only	n/a

*****	asterisk	included only (avoid using * in node names),	Included only	included only (avoid using in parameter names),	Comment in both IBIS-ISS Wildcard character. Double asterisk (**) is “To the power of”.
( )	parentheses	Illegal	Illegal	Illegal	Token delimiter
-	minus	included only	Included only	Illegal	n/a
_	underscore	Legal anywhere	Included only	Included only	n/a
+	plus sign	included only	Included only	included only (avoid usage); Illegal	Continues previous line, except for quoted strings (expressions, paths, algebraics)
=	equals	Illegal	Illegal	optional in .PARAM statements	Token delimiter
< >	less/more than	Legal anywhere	Included only	Included only	n/a
?	question mark	Legal anywhere	Included only	Included only	Wildcard in character in both IBIS-ISS
/	forward slash	Legal anywhere	Included only	Illegal	n/a

{ }	curly braces	included only, converts { } to [ ]	Included only	Included only	Auto-converts to square brackets ([ ])
[ ]	square brackets	Included only	Included only	Included only	n/a
****	backslash (requires a whitespace before to use as a continuation)	included only	Included only	Illegal ~~Illegal~~	Continuation character ~~for quoted strings~~ (preserves whitespace)
****	double backslash ~~(requires a whitespace before to use as a continuation)~~	included only	Illegal	Illegal	Continuation character for quoted strings (preserves whitespace)
|	pipe	Legal anywhere	Included only	Included only	n/a
,	comma	Illegal	Illegal	Illegal	Token delimiter
.	period	Illegal	Included only	Included only	Statement identifier , (i.e., .PARAMETER, etc.).
:	colon	Included only	Included only	Included only	Delimiter for element attributes

;	semi-colon	Included only	Included only	Included only	n/a
" "	double-quotes	Illegal	Illegal	Illegal	Expression and filename delimiter
‘ ’	single quotes	Illegal	Illegal	Illegal	Expression and filename delimiter
	Blank (whitespace)	Use before \ ~~or~~ ~~\\~~ line (token) continuations			Token delimiter

4. First Character

The first character in every line specifies how IBIS-ISS interprets the remainder of the line.

Table 5 *First Character Descriptions*

Line	**If the First Character is...**	**Indicates**
Subsequent lines of netlist, and all lines of included files	. (period)	Statement identifier. For example, .PARAM
	`c, C, e, E, f, F, g, G, h, H, , k, K, l, L, r, R, s, S, v, V, w, W`	Element instantiation
	`*` (asterisk)	Comment line

+ (plus)	Continues previous line

5. Delimiters

Tokens are strings in the input file separated by delimiters. Input token delimiters are: tab, blank, comma (,), equal sign (=), and parentheses ( ).

In addition, single (‘) or double quotes (“) delimit tokens used as expressions and filenames.

6. Instance Names

The names of element instances begin with the element key letter, except for subcircuit instances, whose instance names begin with X. Instance names may be up to 1024 characters long.

Table 6 *Element Identifiers*

Key Letter (First Char)	**Element**	**Example Line**
`C`	Capacitor	`Cbypass 1 0 10pf`
`E`	Voltage-controlled voltage source	`Ea 1 2 3 4 K`
`F`	Current-controlled current source	`Fsub n1 n2 vin 2.0`
`G`	Voltage-controlled current source	`G12 4 0 3 0 10`
`H`	Current-controlled voltage source	`H3 4 5 Vout 2.0`
`K`	Linear mutual inductor (general form)	`K1 L1 L2 1`
`L`	Linear inductor	`LX a b 1e-9`
`R`	Resistor	`R10 21 10 1000`

`S`	S-parameter element	`S1 nd1 nd2 s_model2`
`V`	Voltage source	`V1 8 0 DC=0`
`W`	Transmission Line	`W1 in1 0 out1 0 N=1 L=1`
`T`	Transmission Line	`Txxx in 0 out 0 Z0=50 TD=30n`
`X`	Subcircuit instance	`X1 2 4 17 31 MULTI WN=100 LN=5`

7. Numbers

Numbers may be entered as integer, floating point, floating point with an integer exponent, or integer or floating point with one of the scale factors listed below.

Table 7 *Scale Factors*

Scale Factor	**Prefix**	**Symbol**	**Multiplying Factor**
T	tera	T	1e+12
G	giga	G	1e+9
MEG or X	mega	M	1e+6
K	kilo	k	1e+3
MIL	n/a	none	25.4e-6
M	milli	m	1e-3

U	micro	µ	1e-6
N	nano	n	1e-9
P	pico	p	1e-12
F	femto	f	1e-15
A	atto	a	1e-18

Note:

Scale factor A is not a scale factor in a character string that contains amps. For example, IBIS-ISS interprets the 20amps string as 20e-18mps (20^{-18}amps), but it correctly interprets 20amps as 20 amperes of current, not as 20e-18mps (20^{-18}amps).

- Numbers may use exponential format or engineering key letter format, but not both (1e-12 or 1p, but not 1e-6u).
- To designate exponents, use D or E.
- Trailing alphabetic characters are interpreted as units comments.
- Units comments are not checked.

8. Parameters and Expressions

- Parameter names use IBIS-ISS name syntax rules, except that names must begin with an alphabetic character. The other characters must be either a number, or one of these characters:

```
! # $ % [ ] _
```

- If multiple definitions are given for the same parameter, IBIS-ISS uses the last parameter definition even if that definition occurs later in the input than a reference to the parameter.
- A parameter must be defined before that parameter is used in a definition for another parameter.
- To delimit expressions, use single quotes.
- Expressions cannot exceed 1024 characters.
- Parameters are used in two contexts.
    - Parameters in parameter definition statements are strings, defining names to be used as variables which are assigned

Formatted: Bullet

Formatted: Font: (Default) Helvetica

specific values by the statement. These values may be numeric, strings defining an expression or equation, or strings matching parameters defined elsewhere.

- Parameters may also appear in element instances, model definitions and subcircuit definitions. These parameters may be user-defined or may use names pre-defined by the syntax of the element.

▪ Parameter names must begin with an alphabetic character, but thereafter may contain numbers and/or curly braces ({ }), and/or square brackets ([ ]).

- Parameter names are input tokens. Token delimiters must precede and follow names.
- Parameter names may be up to 1024 characters long and are not case-sensitive.

~~1.~~

~~2. For improved readability, use a double slash (\\) at end of a line, to continue the line.~~

Formatted: Bullet2, Indent: Left: 1", Hanging: 0.29", Bulleted + Level: 1 + Aligned at: 1" + Indent at: 1", Tab stops: Not at 0.5"

Formatted: Heading 2

9. ~~No~~Node Name (or Node Identifier) Conventions

Nodes are the points of connection between elements in the input netlist. Only ~~Either names~~alphanumeric ~~or numbers may~~strings shall be used to designate nodes. If entirely numeric, ~~Node~~ node numbers ~~may be~~shall be between ~~from~~ 1 ~~to~~ and 999999999999999 (1 to 1e16-1~~);~~). ~~node~~ A node number of 0 is permitted but is interpreted as~~always~~ ground. Letters that follow a leading number~~s~~ in a node names are ignored; this means that node strings such as '3n5' and '3' shall be interpreted as referring to the same node.

When the node name begins with a letter or a valid special character, the node name may contain a maximum of 1024 characters.

~~Subcircuit Node Names~~

~~Two subcircuit node names are assigned in this format.~~

To indicate the ground node, use either the number 0, the name `GND`, or `!GND, or GROUND, GND!`. Every node shall have at least two connections, except for transmission line nodes (unterminated transmission lines are permitted) and MOSFET substrate nodes (which have two internal connections).

10. Element, Instance, and Subcircuit Naming Conventions

Formatted: Heading 2

Instances and subcircuits are elements and as such, follow the naming conventions for elements.

Element names begin with a letter designating the element type, followed by up to 1023 alphanumeric characters. Element type letters are R for resistor, C for capacitor and so on.

~~9.~~11. Line Continuations

Lines are continued in one of three ways, depending on how input is divided across the line:

- Statements are continued using the + character
- Tokens are continued using the \\ sequence
- Tokens may also be continued using whitespace followed by the \ character

To continue a statement across multiple lines, the plus (+) sign shall be used as the first non-numeric, non-blank character of each continued line. The + sign shall be used only between tokens and token delimiters and never to split tokens.

~~Line continuations require a plus sign (+) as the first character in the line that follows.~~ Here is an example of comments and line continuation in a netlist file:

```
~~.ABC Title Line~~
~~* on this line, because the first line is always a comment)~~
~~* This is a comment line~~
~~.subckt example n1 n2 $ this is an inline comment~~
~~* This is a comment line~~ *This shows~~and the following line is a~~
continuation of a statement describing a resistor
Rexample
+ n3 n4 R=30
```

To continue a token with extended length (such as paths and expressions) but preserve readability, the token shall be split using a single whitespace character followed by a backslash (\ ) or a double backslash (\\ ) without leading whitespace at the end of the line containing the token to be continued on the following line.

```
* This shows continuation of a token in a statement, in this
* case an expression, describing a resistor
Rexample n3 n4 R='sin(30.12345 + 4.356789 – cos(32.67 -\\
234))'

* This shows an identical continuation of a token in a
* statement, in this case an expression, describing a resistor
Rexample n3 n4 R='sin(30.12345 + 4.356789 – cos(32.67 - \
234))'
```

Formatted: Indent: Left: 0.25", No bullets or numbering, Tab stops: Not at 1"

Formatted: Bulleted + Level: 1 + Aligned at: 0.25" + Indent at: 0.5"

Formatted: Indent: Left: 0", Hanging: 0.31", No bullets or numbering

Formatted: Indent: Left: 0.25", No bullets or numbering

6. Parameters

Parameters are similar to the variables used in most programming languages. Parameters hold values assigned when the circuit design is created or that are calculated based on circuit solution values. Parameters ~~can~~ may store static values for a variety of quantities (resistance, source voltage, rise time, and so on). Parameters may also be alphabetic strings used with elements where string input is expected (for example, filenames or model names).

Using Parameters in Simulation (.PARAM)

Defining Parameters

~~Parameters in IBIS-ISS are strings that are associated with numeric values.~~ Parameters may be defined using the methods shown below.

Table 9 *.PARAM Statement Syntax*

Token	Description
Simple assignment	`.PARAM <SimpleParam>=1e-12`
Algebraic ~~definition~~assignment	`.PARAM <AlgebraicParam>='SimpleParam*8.2'` SimpleParam excludes the output variable.
~~Subcircuit default~~String assignment	.~~SUBCKT~~ PARAM <~~SubName~~StringParam~~>~~ >=~~<ParamDefName>=<Value>~~'mystring' StringParam is assigned the value mystring.
Subcircuit default	`.SUBCKT <SubName> <ParamDefName>=<Value>`

Formatted: Font: (Default) Courier New

Formatted Table

Formatted: Font: (Default) Courier New

Formatted: Font: (Default) Courier New

Formatted: Font: (Default) Courier New

Formatted: Font: (Default) Courier New

Formatted: Font: (Default) Courier New

Formatted: Font: (Default) Courier New

Formatted: Font: (Default) Courier New

Formatted: Font: (Default) Courier New

A parameter definition in IBIS-ISS always uses the last value found in the input netlist. ~~(subject to local versus global parameter rules).~~ The definitions below assign a value of 3 to the *DupParam* parameter.

```
.PARAM DupParam=1
```

```
...
.PARAM DupParam=3
```

IBIS-ISS assigns `3` as the value for all instances of `DupParam`, including instances that are earlier in the input than the `.PARAM DupParam=3` statement.

All parameter values in IBIS-ISS are IEEE double floating point numbers. The parameter resolution order is:

1. Resolve all literal assignments.
2. Resolve all expressions.
3. Resolve all function calls.

(Table) shows the parameter passing order.

Table 10 *Parameter Passing Order*

Formatted Table

.PARAM statement ()	.SUBCKT call (instance)
.SUBCKT call (instance)	.SUBCKT definition (symbol)
.SUBCKT definition (symbol)	.PARAM statement ()

Assigning Parameters

The following types of values may be assigned to parameters:

- Constant real number
- Algebraic expression of real values
- Predefined function
- Circuit value
- Model value
- Strings not for algebraic evaluation

To invoke the algebraic processor, enclose a complex expression in single quotes. A simple expression consists of one parameter name.

The parameter keeps the assigned value, unless a later definition changes its value.

Inline Parameter Assignments

To define circuit values, using a direct algebraic evaluation:

```
r1 n1 0 R='1k/sqrt(HERTZ)' $ Resistance for frequency
```

Using Algebraic Expressions

In IBIS-ISS, an algebraic expression, with quoted strings, ~~can~~ may replace any parameter ~~in the netlist~~.

Some uses of algebraic expressions are:

- Parameters:

```
.PARAM x='y+3'
```

- Algebra in elements:

```
R1 1 0 r='ABS(v(1)/i(m1))+10'
```

In addition to using quotations, the expression inside the `PAR( )` statement must be defined to enable output. The continuation character for quoted parameter strings, in IBIS-ISS, is a double backslash (`\\`)~~.~~ (~~O~~outside of quoted strings, the single backslash (`\`) is the continuation character~~.~~).

Built-In Functions and Variables

In addition to simple arithmetic operations (`+`, `-`, `*`, `/`), the built-in functions and variables listed below may be used in IBIS-ISS expressions.

Table 11 *IBIS-ISS Built-in Functions*

IBIS-ISS Form	Function	Class	Description
sin(x)	sine	trig	Returns the sine of x (radians)
cos(x)	cosine	trig	Returns the cosine of x (radians)

tan(x)	tangent	trig	Returns the tangent of x (radians)
asin(x)	arc sine	trig	Returns the inverse sine of x (radians)
acos(x)	arc cosine	trig	Returns the inverse cosine of x (radians)
atan(x)	arc tangent	trig	Returns the inverse tangent of x (radians)
sinh(x)	hyperbolic sine	trig	Returns the hyperbolic sine of x (radians)
cosh(x)	hyperbolic cosine	trig	Returns the hyperbolic cosine of x (radians)
tanh(x)	hyperbolic tangent	trig	Returns the hyperbolic tangent of x (radians)
abs(x)	absolute value	math	Returns the absolute value of x: $\|x\|$
sqrt(x)	square root	math	Returns the square root of the absolute value of x: sqrt(-x)=-sqrt(|x|)
pow(x,y)	absolute power	math	Returns the value of x raised to the integer part of y: $x^{\text{(integer part of y)}}$
pwr(x,y)	signed power	math	Returns the absolute value of x, raised to the y power, with the sign of x: (sign of x)$\|x\|^y$
x**y	power		If x<0, returns the value of x raised to the integer part of y. If x=0, returns 0. If x>0, returns the value of x raised to the y power.

log(x)	natural logarithm	math	Returns the natural logarithm of the absolute value of x, with the sign of x: (sign of x)log(|x|)
log10(x)	base 10 logarithm	math	Returns the base 10 logarithm of the absolute value of x, with the sign of x: (sign of x)$\log_{10}$(|x|)
exp(x)	exponential	math	Returns e, raised to the power x: e^x
db(x)	decibels	math	Returns the base 10 logarithm of the absolute value of x, multiplied by 20, with the sign of x: (sign of x)20$\log_{10}$(|x|)
int(x)	integer	math	Returns the integer portion of x. The fractional portion of the number is lost.
nint(x)	integer	math	Rounds x up or down, to the nearest integer.
sgn(x)	return sign	math	Returns -1 if x is less than 0. Returns 0 if x is equal to 0. Returns 1 if x is greater than 0
sign(x,y)	transfer sign	math	Returns the absolute value of x, with the sign of y: (sign of y)|x|
def(x)	parameter defined	control	Returns 1 if parameter x is defined. Returns 0 if parameter x is not defined.
min(x,y)	smaller of two args	control	Returns the numeric minimum of x and y
max(x,y)	larger of two args	control	Returns the numeric maximum of x and y

[cond] ?x : y	ternary operator	Returns *x* if *cond* is not zero. Otherwise, returns *y*. .param z='condition ? x:y'
<	relational operator (less than)	Returns 1 if the left operand is less than the right operand. Otherwise, returns 0. .para x=y<z (y less than z)
<=	relational operator (less than or equal)	Returns 1 if the left operand is less than or equal to the right operand. Otherwise, returns 0. .para x=y<=z (y less than or equal to z)
>	relational operator (greater than)	Returns 1 if the left operand is greater than the right operand. Otherwise, returns 0. .para x=y>z (y greater than z)
>=	relational operator (greater than or equal)	Returns 1 if the left operand is greater than or equal to the right operand. Otherwise, returns 0. .para x=y>=z (y greater than or equal to z)
==	equality	Returns 1 if the operands are equal. Otherwise, returns 0. .para x=y==z (y equal to z)
!=	inequality	Returns 1 if the operands are not equal. Otherwise, returns 0. .para x=y!=z (y not equal to z)
&&	Logical AND	Returns 1 if neither operand is zero. Otherwise, returns 0. .para x=y&&z (y AND z)
||	Logical OR	Returns 1 if either or both operands are not zero. Returns 0 only if both operands are zero.

.para x=y||z (y OR z)

Table 12 *IBIS-ISS Special Variables*

IBIS-ISS Form	Function	Class	Description
time	current simulation time	control	Uses parameters to define the current simulation time, during transient analysis.
temper	current circuit temperature	control	Uses parameters to define the current simulation temperature, during transient/temperature analysis.
hertz	current simulation frequency	control	Uses parameters to define the frequency, during AC analysis.

Parameter Scoping and Passing

If parameters are used to define values in sub-circuits, fewer similar cells should be used, to provide enough functionality in the resulting library. Circuit parameters may be passed into hierarchical designs, and different values may be assigned to the same parameter within individual cells, when simulations are run.

A parameter is defined either by a .parameter statement (local to that subcircuit), or may be passed into a subcircuit, or may be defined on a .subckt definition line.

~~(Some details need to be clarified on this)~~

```
.param x=0
.subckt def
.param x=1
x1 1 2 abc x=2
.subckt abc 1 2 x=3
.param x=3
r1 1 2 R=x
.ends abc
.ends def
.end
```

Formatted: Font: (Default) Courier New, 11 pt

Formatted: Font: (Default) Courier New, 11 pt, Not Bold

Formatted: Font: (Default) Courier New, 11 pt

Formatted: Font: (Default) Courier New, 11 pt, Not Bold

Formatted: Font: (Default) Courier New, 11 pt

Formatted: Font: (Default) Courier New, 11 pt, Not Bold

Formatted: Font: (Default) Courier New, 11 pt

Formatted: Font: (Default) Courier New, 11 pt

Formatted: Font: (Default) Courier New, 11 pt

Formatted: Font: (Default) Courier New, 11 pt, Not Bold

Formatted: Font: (Default) Courier New, 11 pt, Not Bold

Formatted: Font: (Default) Courier New, 11 pt

The specific details of any particular hierarchy are left to the choice of the user.

This section describes the scope of parameter names, and how IBIS-ISS resolves naming conflicts between levels of hierarchy.

Library Integrity ~~(Needs careful discussion)~~

Integrity is a fundamental requirement for any symbol library. Library integrity can be as simple as a consistent, intuitive name scheme, or as complex as libraries with built-in range checking.

Library integrity might be poor libraries from different vendors are used in a single circuit design. Because names of circuit parameters are not standardized between vendors, two components can include the same parameter name for different functions. For example, one vendor might build a library that uses the name `Tau` as a parameter to control one or more subcircuits in their library. Another vendor might use `Tau` to control a different aspect of their library. If a global parameter named `Tau` is used to control one library, the behavior of the second library may be unintentionally affected.. This is why Best Practices recommends that Global Parameters be avoided.

7. File Includes

The ~~i~~Include statement inserts another file's contents~~-~~ in the current file at evaluation.

Syntax

`.INCLUDE ‘file_path file_name’`

`.inc ‘file_path file_name’`

Arguments

Argument	Description
file_path	Path name of a file for computer operating systems that support tree-structured directories. An include file can contain nested .INCLUDE calls to itself or to another include file. If a relative path is used in a nested .INCLUDE call, the path starts from the directory of the parent .INCLUDE file, not from the current working directory. If the path starts from the current working directory, IBIS-ISS may also find the .INCLUDE file, but prints a warning.
file_name	Name of a file to include in the data file. The file path, plus the file name, may be up to 16 characters long. Any name valid under the computer's operating system may be used.

~~Description~~

Use this ~~command~~ token to include another netlist in the current ~~netlist~~circuit description. A ~~netlist may be used as a subcircuit in one or more other netlists.~~ The file path and file name shall be enclosed in single or double quotation marks.

```
.INCLUDE `/myhome/subcircuits/diode_circuit´
```

8. Comments ~~and Line Continuation~~

Comments require an asterisk (*) as the first character in a line or a dollar sign ($) directly in front of the comment anywhere on the line. For example:

```
* <comment_on_a_line_by_itself>
```

or

```
<IBIS-ISS statement> $ <comment following input>
```

Comment statements may appear anywhere in the circuit description. The dollar sign ($) must be used for comments that do *not* begin in the first character position on a line (for example, for comments that follow simulator input on the same line). If it is not the first nonblank character, then the dollar sign must be preceded by either:

Whitespace
Comma (,)
Valid numeric expression

The dollar sign may also be used within node or element names. For example:

```
* RF=1K GAIN SHOULD BE 100
$ CIRCUIT EXAMPLE
VIN 1 0 PL 0 0 5V 5NS $ 10v 50ns
R12 1 0 1MEG $ FEED BACK
.PARAM a=1w$comment a=1, w treated as a space and ignored
.PARAM a=1k$comment a=1e3, k is a scale factor
```

A dollar sign is the preferred way to indicate comments, because of the flexibility of its placement within the code.

~~Line continuations require a plus sign (+) as the first character in the line that follows. Here is an example of comments and line continuation in a netlist file:~~

~~.ABC Title Line~~
~~* on this line, because the first line is always a comment)~~
~~* This is a comment line~~
~~.MODEL n1 NMOS $ this is an example of an inline comment~~
~~*** This is a comment line and the following line is a continuation**~~

Formatted: Font: (Default) Courier New, Not Italic, Character scale: 100%
Formatted: Font: Courier New
Formatted: Font: (Default) Courier New, Not Italic, Character scale: 100%
Formatted: Font: Courier New
Formatted: Font: (Default) Courier New, Not Italic, Character scale: 100%
Formatted: Font: Courier New
Formatted: Font: (Default) Courier New, Not Italic, Character scale: 100%
Formatted: Font: Courier New
Formatted: Font: (Default) Courier New, Bold, Not Italic, Character scale: 0%

Model Definitions (.MODEL Statements)

Model definitions are used to specify the electrical parameters for W-element and S-element instances. They can be considered a special form of subcircuit definition, in which the defined subcircuit is only available to W- and S-elements.

The specific syntax for W-element and S-element .MODEL definitions are detailed below, as part of the W-element and S-element portions of the IBIS-ISS specification. Note that .MODEL statements are hierarchically at the same level as element instances.

9. Subcircuit Definitions

Syntax

```
.subckt name n1 n2…
statement
statement
statement
…
.ends
```

1. ~~Subckt~~ Subcircuit Scoping Rules

A .subckt or .model definition must occur in the subc~~k~~ircuit in which the subc~~k~~ircuit or model is referenced, or in a calling ~~subckt~~ subcircuit at any level above.

10. Subcircuit Definition Ending Statements

Subcircuit definitions must be ended with the .ends token. See Subcircuit Definitions above for syntax and examples.

11. Elements

1. Subcircuits

Using Subcircuits

Reusable cells are the key to saving labor in any CAD system. This also applies to circuit simulation, in IBIS-ISS

- To create and simulate a reusable circuit, construct it as a subcircuit.
- Use parameters to expand the utility of a subcircuit.

`X`<*subcircuit_name*> creates an instance of a subcircuit. . The subcircuit must have already been defined elsewhere in the IBIS-ISS file using a `.SUBCKT` command.

Syntax

```
Xxxxx n1 [n2 n3 …] subnam
[parnam = val] [M = val]
```

Argument	Definition
X<*subcircuit_name*>	Subcircuit element name. Must begin with an X, followed by up to 15 alphanumeric characters.
n1 …	Node names for external reference.
subnam	Subcircuit model reference name.
Parnam	A parameter name set to a value (val) for use only in the subcircuit. It overrides a parameter value in the subcircuit definition, but is overridden by a value set in a `.PARAM` statement.
M	Multiplier

2. Linear Resistor

Syntax

`R`*`xxx node1 node2`* `[R =]` *`value`*

The value of a linear resistor may be a constant, or an expression of parameters.

Token	Description
Rxxx	Name of a resistor
node1 and node2	Names or numbers of the connecting nodes
value	resistance value, in ohms

3. Linear Capacitor

Syntax

`C`*`xxx node1 node2`* `[C=]`*`value`*

The value of a linear capacitor may be a constant, or an expression of parameters.

Token	Description
C*xxx*	Name of a capacitor. Must begin with C, followed by up to 1023 alphanumeric characters.
node1,node2	Names of connecting nodes.
value	Capacitance value, in farads.

4. Voltage Shunt

A voltage shunt creates a short between two nodes.

Syntax

`V`*`xxx node1 node2`* `[DC=]0`

Token	Description
V*xxx*	Voltage shunt element name. Must begin with K, followed by up to 1023 alphanumeric characters.
node1, node 2	Nodes between which the shunt is placed.
DC=0	The zero value is required and sets the voltage between the nodes at zero volts. The text "DC=" is optional.

5. Mutual Inductor

A mutual inductor describes inductive coupling between two defined inductors.

Syntax

`K`*`xxx`* `L`*`yyy`* `L`*`zzz`* `[K=]`*`coupling`*

Token	Description
Kxxx	Mutual inductor element name. Must begin with K, followed by up to 1023 alphanumeric characters.
Lyyy	Name of the first of two coupled inductors. This inductor must be defined elsewhere in the file.
Lzzz	Name of the second of two coupled inductors. This inductor must be defined elsewhere in the file.
coupling	Coefficient of mutual coupling. This is a unitless number, with magnitude > 0. If the coupling coefficient is negative, the direction of coupling reverses. This is equivalent to reversing the polarity of either of the coupled inductors. Use the K=*xxx* syntax when defining the coupling coefficient using a parameter name or an equation. The pre-defined parameter "k" may be

omitted.

6. Linear Inductor

Syntax

L*xxx node1 node2* [L =] *inductance*

Token	**Description**
Lxxx	Name of an inductor.
node1,node2	Names or numbers of the connecting nodes.
inductance	inductance value, in henries.

7. T-element (Ideal Transmission Line)

Syntax

T*xxx in refin out refout* Z0=*val* TD=*val [*L=*val]*

+ *[*IC=*v1, i1, v2, i2]*

Token	Description
Txxx	Lossless transmission line element name. Must begin with T, followed by up to 1023 alphanumeric characters.
In	Signal input node.

Refin	Ground reference for the input signal.
Out	Signal output node.
Refout	Ground reference for the output signal.
Z0	Characteristic impedance of the transmission line (Ohms).
TD	Propagation time delay of the transmission line (in seconds). If physical length (L) is specified, then units for TD are considered in seconds per meter.
L	Physical length of the transmission line, in units of meters. Default=1.

8. W-element (Coupled Transmission Line)

~~Describes~~ This section describes how to use basic transmission line simulation equations and an optional method for computing the parameters of transmission line equations.

Formatted: Font: Not Italic

Formatted: Font: Not Italic

The W-element is a versatile transmission line model that may be used to describe a variety of transmission line structures, from a simple lossless line to complex frequency-dependent lossy-coupled lines.

Syntax

```
Wxxx i1 i2 ... iN iR o1 o2 ... oN oR N=val L=val
+ [RLGCMODEL=name | TABLEMODEL=name]
```

Token	Description
N	Number of signal conductors (excluding the reference conductor).
i1...iN	Node names for the near-end signal-conductor terminal
iR	Node name for the near-end reference-conductor terminal.
o1... oN	Node names for the far-end signal-conductor terminal
oR	Node name for the far-end reference-conductor terminal.
L	Length of the transmission line.
RLGCMODEL	Name of the RLGC model.
TABLEMODEL	Name of the frequency-dependent tabular model

The W-element supports two formats to specify transmission line properties:

- Format 1: RLGC specification
  - Internally specified in a `.MODEL` statement.
  - Externally specified in a different file.
- Format 2: Frequency-dependent tabular specification

Parameters in the W-element element declaration may be declared in any order. Specify the number of signal conductors, N, after the list of nodes. The nodes and parameters in the W-element element declaration may be interspersed.

Format 1: RLGC Model

Equations and Parameters on page 96 ~~(NOTE: Do we want to include these explanations)~~ describes the inputs of the W-element per unit length matrices: R_o (DC resistance), L, G, C, R_s (skin effect), and G_d (dielectric loss)

The W-element does not limit any of the following parameters:

- Number of coupled conductors.
- Shape of the matrices.
- Line loss.
- Length or amount of frequency dependence.

The RLGC text file contains frequency-dependent RLGC matrices per unit length. The W-element also handles frequency-independent RLGC, and lossless (LC) lines. It does not support RC lines.

Because RLGC matrices are symmetrical, the RLGC model specifies only the lower triangular parts of the matrices. The syntax of the RLGC model for the W-element is:

```
.MODEL name W MODELTYPE=RLGC N=val
+ Lo=matrix_entries
+ Co=matrix_entries [Ro=matrix_entries Go=matrix_entries]
+ Rs=matrix_entries wp=val Gd=matrix_entries Rognd=val
+ Rsgnd=val Lgnd=val
```

Token	Description
N	Number of conductors (same as in the element card).

L

DC inductance matrix, per unit length $\left[\frac{H}{m}\right]$.

C

DC capacitance matrix, per unit length $\left[\frac{F}{m}\right]$.

Ro

DC resistance matrix, per unit length $\left[\frac{\Omega}{m}\right]$.

Go

DC shunt conductance matrix, per unit length $\left[\frac{S}{m}\right]$.

Rs

Skin effect resistance matrix, per unit length $\left[\frac{\Omega}{m\sqrt{Hz}}\right]$.

Gd

Dielectric loss conductance matrix, per unit length $\left[\frac{S}{m \cdot Hz}\right]$.

wp

Angular frequency of the polarization constant [radian/sec] (see Introduction to the Complex Dielectric Loss Model on page 99)). When the wp value is specified, the unit of Gd becomes [S/m].

Lgnd

DC inductance value, per unit length for grounds $\left[\frac{H}{m}\right]$ (reference line).

Rognd

DC resistance value, per unit length for ground $\left[\frac{\Omega}{m}\right]$.

Rsgnd

Skin effect resistance value, per unit length for ground $\left[\frac{\Omega}{m\sqrt{Hz}}\right]$.

The following input netlist file shows RLGC input for the W-element:

```
* W-Element example, four-conductor line
```

```
W1 N=3 1 3 5 0 2 4 6 0 RLGCMODEL=example_rlc l=0.97
```

```
* RLGC matrices for a four-conductor lossy
.MODEL example_rlc W MODELTYPE=RLGC N=3
+ Lo=
+ 2.311e-6
+ 4.14e-7 2.988e-6
+ 8.42e-8 5.27e-7 2.813e-6
+ Co=
+ 2.392e-11
+ -5.41e-12 2.123e-11
+ -1.08e-12 -5.72e-12 2.447e-11
+ Ro=
+ 42.5
+ 0 41.0 + 0 0 33.5
+ Go= + 0.000609
+ -0.0001419 0.000599
+ -0.00002323 -0.00009 0.000502
+ Rs=
+ 0.00135
+ 0 0.001303
+ 0 0 0.001064
+ Gd=
+ 5.242e-13
+ -1.221e-13 5.164e-13
+ -1.999e-14 -7.747e-14 4.321e-13
```

Using RLGC Matrices

RLGC matrices in the RLGC model of the W-element are in the Maxwellian format

Format 2: Frequency-Dependent Tabular Specification

The tabular RLGC model may be used as an extension of the analytical RLGC model to model any arbitrary frequency-dependent behavior of transmission lines (this model does not support RC lines).

The W-element syntax supports tables of data (use a `.MODEL` statement of type w). To accomplish this, the `.MODEL` statement refers to .MODEL statements where the “type” is SP (described in Small-Signal Parameter Data Frequency Table Model (SP Model) on page 77), which contain the actual table data for the RLGC matrices.

Note:

To ensure accuracy, the W-element tabular model requires the following:

- R and G tables require zero frequency points.
- L and C tables require infinity frequency points as well as zero frequency points.

To specify a zero frequency point, the pre-defined DC parameter may be used. Alternatively, the f parameter in the DATA field of the SP model may be set to a value of 0. To specify an infinity frequency point, use the INFINITY token.

See also, Small-Signal Parameter Data Frequency Table Model (SP Model) on page 77.

W-element Model Definition Syntax

```
.MODEL name W MODELTYPE=TABLE [FITGC=0|1] N=val
+ LMODEL=l_freq_model CMODEL=c_freq_model
+ [RMODEL=r_freq_model GMODEL=g_freq_model]
```

Token	**Description**
FITCG	Pre-defined parameter token for the W-element with MODELTYPE=TABLE. A value of 1 instructs the tool to run a causality check on the data. A value of 0 turns any causality checking off (default)
N	Number of signal conductors (excluding the reference conductor).
LMODEL	SP model name for the inductance matrix array.
CMODEL	SP model name for the capacitance matrix array.
RLMODEL	SP model name for the resistance matrix array. By default, it is zero.
GMODEL	SP model name for the conductance matrix array. By default, it is zero.

9. S-element

An S-element is a frequency-domain set of network data, described using scattering parameters.

Syntax

```
Sxxx nd1 nd2 ... ndN [ndRef]

+ MNAME=Smodel_name

+ [FBASE = base_frequency] [FMAX=maximum_frequency]
```

Token	Description
nd1 nd2...ndN	Nodes of an S-element Three kinds of definitions are present: ▪ With no reference node ndRef, the default reference node is GND. Each node ndi (i=1~N) and GND construct one of the N ports of the S-element. ▪ With one reference node, ndRef is defined. Each node ndi (i=1~N) and the ndRef construct one of the N ports of the S-element. ▪ With an N reference node, each port has its own reference node. The node definition may be written more clearly: nd1+ nd1- nd2+ nd2- ... ndN+ ndN- Each pair of the nodes (ndi+ and ndi-, i=1~N) constructs one of the N ports of the S-element.
ndRef	Reference node
MNAME	Name of the S model; Note that string parameters are supported in calling an MNAME.

FBASE

Base frequency to use for transient analysis. This value becomes the base frequency point for Inverse Fast Fourier Transformation (IFFT).

- If this value is not set, the base frequency is a reciprocal value of the transient period.
- If you set a frequency that is smaller than the reciprocal value of the transient, then transient analysis performs circular convolution, and uses the reciprocal value of FBASE as its base period.

FMAX

Maximum frequency use in transient analysis. Used as the maximum frequency point for Inverse Fast Fourier Transformation (IFFT).

The nodes of the S-element must come first. You can specify all the optional parameters in both the
S-element and S model statements, except for `MNAME` argument.

The optional arguments may be entered in any order, and the parameters specified in the element statement have a higher priority.

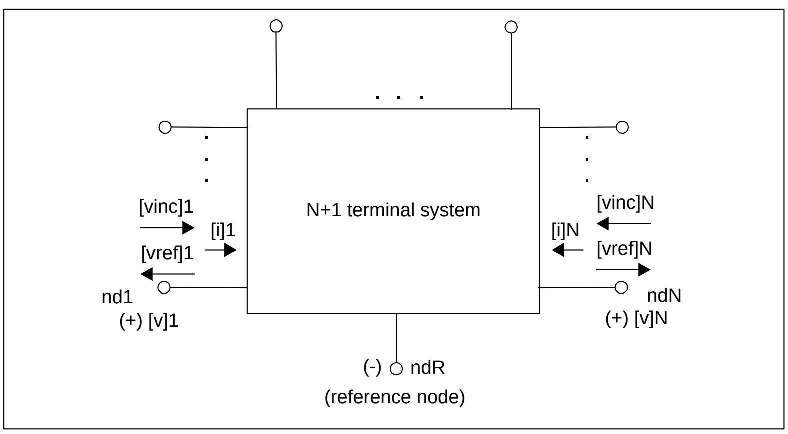

Figure 27 *Terminal Node Notation*

Node Example

The following example illustrates the *nd1 nd2...ndN—no reference*, *single reference*, and multi-reference parameters.

```
**S-parameter example
```

```
* no reference
S_no_ref n1 n2 mname=s_model

* single reference
S_one_ref n1 n3 gnd mname=s_model

*multi-reference
S_multi_ref n1 gnd n4 gnd mname=s_model
```

The S-element must have a call to one of the supported S-parameter file formats (IBIS-ISS gets the number of ports from the S-parameter file The number of ports, ‘n’, may be specified explicitly as N=n.

- For n terminals, the S-element assumes no reference node.
- For n+1 terminals, the S-element assumes one reference node.
- For 2n terminals, the S-element assumes signal nodes and n reference nodes. Each pair of nodes is a signal and a reference node.

S Model Syntax

Use the following syntax to describe specific S models:

```
.MODEL Smodel_name S [N=dimension]
+ [TSTONEFILE=filename

+ [FBASE=base_frequency] [FMAX=maximum_frequency]
```

Token	Description
Smodel_name	Name of the S model.
S	Specifies that the model type is an S model.
N	S model dimension, which is equal to the terminal number of an S-element and excludes the reference node.

TSTONEFILE	Specifies the name of a Touchstone file. Data contains frequency-dependent array of matrixes. Touchstone files must follow the .s#p file extension rule, where # represents the dimension of the network. Note that string parameters are supported for TSTONEFILE Example: .subckt sparam n1 n2 tsfile=str('ss_ts.s2p') S1 n1 n2 0 mname=s_model .model s_model S TSTONEFILE=str(tsfile) .ends x1 A B sparam tsfile=str('ss_ts.s2p') … For details, see *Touchstone® File Format Specification* by the EIA/IBIS Open Forum (http://www.eda.org).
FBASE	Base frequency used for transient analysis. IBIS-ISS uses this value as the base frequency point for Fast Inverse Fourier Transformation (IFFT). ▪ If FBASE is not set, IBIS-ISS uses a reciprocal of the transient period as the base frequency. ▪ If FBASE is set smaller than the reciprocal value of transient period, transient analysis performs circular convolution by using the reciprocal value of FBASE as a base period.
FMAX	Maximum frequency for transient analysis. Used as the maximum frequency point for Inverse Fast Fourier Transform (IFFT).

The, `TSTONEFILE` parameters describe the frequency-varying behavior of a network.

10. E-element (Voltage-Controlled Voltage Source)

This section explains the E-element syntax and parameters.

Linear

```
Exxx n+ n- [VCVS] in+ in- gain
```

For a description of these parameters, see table VCVS Parameters.

Laplace Transform

Voltage Gain H(s):

```
Exxx n+ n- LAPLACE in+ in-   k0, k1, ..., kn / d0,
   d1, ..., dm
```

For a description of these parameters, see table VCVS Parameters.

H(s) is a rational function, with parameters used to define the values of all coefficients (k_0, k_1, ..., d_0, d_1, ...).

Pole-Zero Function

Voltage Gain H(s):

```
Exxx n+ n- POLE in+ in- a az1, fz1, ..., azn, fzn / b,
+ ap1, fp1, ..., apm, fpm
```

For a description of these parameters, see table VCVS Parameters.

The following equation defines H(s) in terms of poles and zeros:

$$H(s) = \frac{a \cdot (s + \alpha_{z1} - j2\pi f_{z1}) \ldots (s + \alpha_{zn} - j2\pi f_{zn})(s + \alpha_{zn} + j2\pi f_{zn})}{b \cdot (s + \alpha_{p1} - j2\pi f_{p1}) \ldots (s + \alpha_{pm} - j2\pi f_{pm})(s + \alpha_{pm} + j2\pi f_{pm})}$$

The complex poles or zeros are in conjugate pairs. The element description specifies only one of them, and the program includes the conjugate. Parameters may be used to specify the a, b, α, and f values.

Example

```
Elow_pass out 0 POLE in 0 1.0 / 1.0, 1.0,0.0 0.5,0.1379
```

The `Elow_pass` statement describes a low-pass filter, with the transfer function:

$$H(s) = \frac{1.0}{1.0 \cdot (s+1)(s+0.5+j2\pi \cdot 0.1379)(s+0.5-(j2\pi \cdot 0.1379))}$$

Foster Pole-Residue Form

Gain E(s) form

```
Exxx n+ n- FOSTER in+ in- k0 k1
+ (Re{A1}, Im{A1})/ (Re{p1}, Im{p1})
+ (Re{A2}, Im{A2})/ (Re{p2}, Im{p2})
+ (Re{A3}, Im{A3})/ (Re{p3}, Im{p3})
+ ...
```

For a description of these parameters, see table VCVS Parameters.

In the above syntax, parenthesis , commas, and slashes are separators—they have the same meaning as a space. A pole-residue pair is represented by four numbers (real and imaginary part of the residue, then real and imaginary part of the pole).

You must make sure that Re[pi]<0; otherwise, the simulations will certainly diverge. Also, it is a good idea to assure passivity of the model (for an N-port admittance matrix Y, Re{Y} should be positive-definite), or the simulation is likely to diverge).

Note:

For real poles, half the residue value is entered because it is applied twice. In the above example, the first pole-residue pair is real, but is written as “A1/(s-p1)+A1/(s-p1)”; therefore, 0.0004 is entered rather than 0.0008.

Table VCVS Parameters.

E-element Parameters

The E-element parameters are described in the following list.

Token	Description
Exxx	Voltage-controlled element name. Must begin with E, followed by up to 1023 alphanumeric characters.
gain	Voltage gain.
in +/-	Positive or negative controlling nodes. Specify one pair for each dimension.
K	Ideal transformer turn ratio: $\mathrm{V(in+,in-)} = k \cdot \mathrm{V(n+,n-)}$ or, number of gates input.
n+/-	Positive or negative node of a controlled element.
VCVS	Pre-defined token for a voltage-controlled voltage source. VCVS is a reserved word; do not use it as a node name.

11. F-element (Current-Controlled Current Source)

This section explains the F-element syntax and parameters.

Note:

G-elements with algebraic expressions may be used to duplicate the functions of an F-element.

Formatted: NotePara

Syntax

```
Fxxx n+ n- [CCCS] vn1 gain
```

F-element Parameters

The F-element parameters are described in the following list.

Token	Description
CCCS	Pre-defined token for current-controlled current source. CCCS is a IBIS-ISS reserved word; do not use it as a node name.
Fxxx	Element name of the current-controlled current source. Must begin with F, followed by up to 1023 alphanumeric characters.
gain	Current gain.
n+/-	Connecting nodes for a positive or negative controlled source.
vn1 …	Names of voltage sources, through which the controlling current flows. Specify one name for each dimension.
x1,...	Controlling current, through the *vn1* source. Specify the *x* values in increasing order.
y1,...	Corresponding output current values of *x*.

12. G-element (Voltage-Controlled Current Source)

This section explains G-element syntax statements, and their parameters.

Linear

```
Gxxx n+ n- [VCCS] in+ in- transconductance
```

For a description of the G-element parameters, see Table VCCS Parameters.

Laplace Transform

Transconductance H(s):

```
Gxxx n+ n- LAPLACE in+ in-   k0, k1, ..., kn / d0,
   d1, ..., dm
```

H(s) is a rational function, in the following form:

$$H(s) = \frac{k_0 + k_1 s + \ldots + k_n s^n}{d_0 + d_1 s + \ldots + d_m s^m}$$

Parameters may be used to define the values of all coefficients (k_0, k_1, ..., d_0, d_1, ...).

Pole-Zero Function

Transconductance H(s):

```
Gxxx n+ n- POLE in+ in- a az1, fz1, ..., azn, fzn / b,
+ ap1, fp1, ..., apm, fpm
```

The following equation defines H(s) in terms of poles and zeros:

$$H(s) = \frac{a \cdot (s + \alpha_{z1} - j2\pi f_{z1}) \ldots (s + \alpha_{zn} - j2\pi f_{zn})(s + \alpha_{zn} + j2\pi f_{zn})}{b \cdot (s + \alpha_{p1} - j2\pi f_{p1}) \ldots (s + \alpha_{pm} - j2\pi f_{pm})(s + \alpha_{pm} + j2\pi f_{pm})}$$

The complex poles or zeros are in conjugate pairs. The element description specifies only one of them, and the program includes the conjugate. You can use parameters to specify the a, b, α, and f values.

For a description of the G-element parameters, see table VCVS Parameters.

Example

```
Ghigh_pass 0 out POLE in 0 1.0 0.0,0.0 / 1.0 0.001,0.0
```

The `Ghigh_pass` statement describes a high-pass filter, with the transfer function:

$$H(s) = \frac{1.0 \cdot (s + 0.0 + j \cdot 0.0)}{1.0 \cdot (s + 0.001 + j \cdot 0.0)}$$

Foster Pole-Residue Form

Transconductance G(s) form

```
Gxxx n+ n- FOSTER in+ in- k0 k1
+ (Re{A1}, Im{A1})/ (Re{p1}, Im{p1})
+ (Re{A2}, Im{A2})/ (Re{p2}, Im{p2})
+ (Re{A3}, Im{A3})/ (Re{p3}, Im{p3})
+ ...
```

In the above syntax, parenthesis , commas, and slashes are separators—they have the same meaning as a space. A pole-residue pair is represented by four numbers (real and imaginary part of the residue, then real and imaginary part of the pole).

You must make sure that Re[pi]<0; otherwise, the simulations will certainly diverge. Also, it is a good idea to assure passivity of the model (for an N-port admittance matrix Y, Re{Y} should be positive-definite), or the simulation is likely to diverge).

For a description of the G-element parameters, see table VCVS Parameters.

Example

To represent a G(s) in the form,

$$G(s) = 0.001 + 1 \times 10^{-12}s + \frac{0.0008}{s + 1 \times 10^{10}} + \frac{(0.001 - j0.006)}{s - (-1 \times 10^{8} + j1.8 \times 10^{10})} + \frac{(0.001 + j0.006)}{s - (-1 \times 10^{8} - j1.8 \times 10^{10})}$$

The IBIS-ISS syntax would be

```
G1 1 0 FOSTER 2 0 0.001 1e-12
+(0.0004, 0)/(-1e10, 0) (0.001, -0.006)/(-1e8, 1.8e10)
```

Note:

For real poles, half the residue value is entered because it is applied twice. In the above example, the first pole-residue pair is real, but is written as “A1/(s-p1)+A1/(s-p1)”; therefore, 0.0004 is entered rather than 0.0008.

G-element Parameters

The G-element parameters described in the following list.

Token	Description
Gxxx	Name of the voltage-controlled element. Must begin with G, followed by up to 1023 alphanumeric characters.
in +/-	Positive or negative controlling nodes. Specify one pair for each dimension.
n+/-	Positive or negative node of the controlled element.
transconductance	Voltage-to-current conversion factor.
VCCS	Pre-defined token for the voltage-controlled current source. VCCS is a reserved IBIS-ISS word; do not use it as a node name.
x1,...	Controlling voltage, across the *in*+ and *in*- nodes. Specify the *x* values in increasing order.
y1,...	Corresponding element values of *x*.

13. H-element (Current-Controlled Voltage Source)

This section explains H-element syntax statements, and defines their parameters.

Note:

The E-element with algebraic expressions may be used to duplicate the function of the H-element.

Syntax

```
Hxxx n+ n- [CCVS] vn1 transresistance
```

Token	Description
CCVS	Pre-defined token for the current-controlled voltage source. CCVS is a IBIS-ISS reserved word; do not use it as a node name.
Hxxx	Element name of current-controlled voltage source. Must start with H, followed by up to 1023 alphanumeric characters.
n+/-	Connecting nodes for positive or negative controlled source.
transresistance	Current-to-voltage conversion factor.
vn1 …	Names of voltage sources, through which controlling current flows. You must specify one name for each dimension.
x1,...	Controlling current, through the *vn1* source. Specify the *x* values in increasing order.
y1,...	Corresponding output voltage values of *x*.